DOI No.: 10.5963/IJMT0102002
International Journal of Multimedia Technology **IJMT**

Low-Cost Real-Time Stereo Vision Hardware with Binary Confidence Metric and Disparity Refinement

Andy Motten[1], Luc Claesen[2]
Expertise Centre for Digital Media, Hasselt University – tUL - IBBT
Wetenschapspark 2, 3590 Diepenbeek, Belgium
[1]andy.motten@uhasselt.be,[2]luc.claesen@uhasselt.be

Abstract-**This paper presents a real-time stereo vision System-on-Chip (SoC) architecture for a depth-field generation processor as required in 3D TV applications. Dense Stereo Vision is a complex problem and uses many resources to achieve a high quality depth image. To reduce the needed resources, this architecture divides the problem into two parts: first a rough depth map is constructed using a segmentation based SAD window comparison, second a disparity refinement module identifies false matches and replaces them with new estimates. The architecture has been constructed to reduce memory usage. There is no need for external memories. It has the additional advantage that the latency between the input frame and output frame becomes minimal. The current implementation is tailored to accommodate low resource devices like the Altera Cyclone-II series. A real-time stereo matching calculation at a frame rate of 56 Hz with a resolution of 800x600 and a disparity of 80 has been realized using this architecture. A Matlab based framework has been realized that automates the simulation and hardware generation of this architecture.**

Keywords-real-time stereo matching; low-cost; real-time; confidence metric, adaptable window; computer vision; Parallel memory architecture; system-on-chip; FPGA; 3D-TV; depth image

I. INTRODUCTION

Stereo matching has long been an important research topic in computer vision and graphics. Many stereo matching algorithms have been investigated and published. A good comparison between different algorithms can be found in the review papers of Lazaros et al. [1], and of Scharstein and Szeliski [2]. Dense stereo matching algorithms can be divided in local (area-based) and global (energy-based) algorithms.

Implementations on hardware are mainly based on a local window based stereo matching architecture. Local and window based methods calculate the differences between the left and the right image from a small part of the images. They produce a decent depth image result and are suitable for real-time applications. The matching cost computation often used is the sum of absolute differences (SAD) [1,2] or the census transform [3]. The support window can be square, rectangular or adaptable.

The adaptive-weight algorithm proposed by Yoon and Kweon [4] adjusts the support weight of each pixel in a fixed sized window. The support weights are depending on the color and spatial difference between each pixel in the window and the centre pixel. Dissimilarities are computed based on the support weights and the plain similarity scores. Their experiment indicates that a local based stereo matching algorithm can produce high quality depth maps similar to global algorithms. A hardware implementation of this method is published by Motten and Claesen [5].

The confidence measurement indicates for each depth value a correctness confidence, which is important for post-processing operations. The cost metric primarily used, is the "left right consistency test" [6, 7]. A good comparison between different cost metrics can be found in the evaluation paper of Hu and Mordohai [8]. However they do not incorporate cost metrics constructed from the neighbouring pixels. Confidence measures suitable for hardware implementation can be found in [9]. They conclude that neighbouring pixels contain valuable information to distinguish good matches from bad matches.

This paper presents a real-time stereo matching System-on-Chip (SoC) architecture for a depth-field generation processor as required in 3D TV applications. The pixels coming from the camera first pass a pre-processing module consisting of a median filter, demosaicing unit and a rectification unit. Next, the two input streams are compared using a segmentation based SAD "Winner Takes All" (WTA) algorithm, which outputs the depth value. Lastly, this depth value is passed into the post-processing module where a binary confidence metric is calculated, which indicates mismatched pixels and calculates new depth values from the neighbouring depth values.

The current implementation of this architecture allows for a real-time stereo matching calculation at a frame rate of 56 Hz with a resolution of 800x600 and a disparity of 80 on a single FPGA without the need for external memories. This architecture is fully scalable and parameterizable to allow for custom SoC implementations, as well as rapid prototyping on FPGAs.

The remainder of the paper is organized as follows: Section two describes related hardware implementations. Section three presents an implementation of a stereo matching architecture using the proposed architecture. Section four discusses the hardware implementation and section five draws the conclusions.

II. RELATED WORK

Recently many stereo implementations have been proposed for hardware implementation. This section will briefly discuss a selection of them.

Murphy et al. [10] proposes a low cost stereo vision system on an FPGA, based on the census transform algorithm. A window-parallel pixel-serial architecture-based VLSI processor to maximize the utilization percentage of processing elements with an adaptable window is presented by Hariyama and Kameyama [11]. Ambrosh et al. [6] proposes an FPGA based stereo matching architecture that uses SAD and a tree based minima calculation. The architecture is highly pipelined, which allows to achieve a frame rate of 600 fps using 450 x 375 input images with a disparity of 150 pixels. Yi et al. [7] compares the resource requirements and performance on a FPGA of an SAD based stereo matching implementation. Different shapes of windows are evaluated to check their influence on the generated depth image. A reduction of the amount of adders used is achieved by decomposing the block SAD calculation in a column and a row SAD calculation. A real-time FPGA-based stereo vision system is presented by Jin et al. [12] that makes use of the census transform. Their system includes all the pre- and post-processing functions as rectification, LR-check and uniqueness test in a single FPGA.

To the author's knowledge, this paper presents the first hardware implementation which features an advanced stereo vision system completely implemented in a single FPGA without the need for external memories.

III. STEREO MATCHING ARCHITECTURE

A. *Basics and Requirements*

The stereo matching algorithm takes two undistorted and rectified images that have been taken by two cameras that have a vertical alignment and a horizontal offset (Fig. 1). Objects will appear on both images on the same horizontal line (the epipolar line). The horizontal distance between the same objects on the left and right images is called the disparity [2]. Objects that are close to the cameras will have a larger horizontal disparity than objects that are far away. The goal of the stereo matching algorithm is to measure the disparity between all pixels in the left and the right image.

The main advantage of dedicated VLSI or SoC architectures in comparison to general purpose CPUs or GPUs is its inherent parallelism and independence of architecture. The major focus of the architecture presented in this paper is on the maximization of the parallel calculations needed for stereo vision processing.

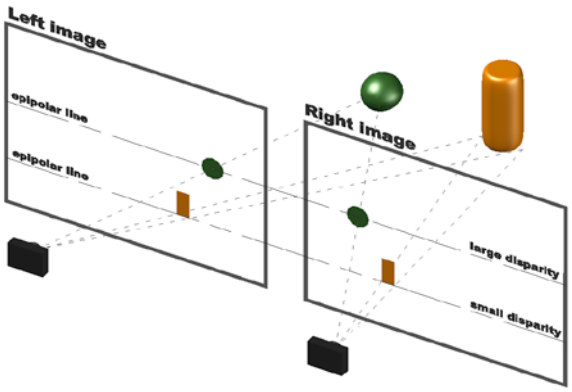


Fig. 1 Stereo vision setup

B. *General Structure*

The global architecture consists of two input streams from two cameras, one window comparison module and one post-processing module (Fig. 2).

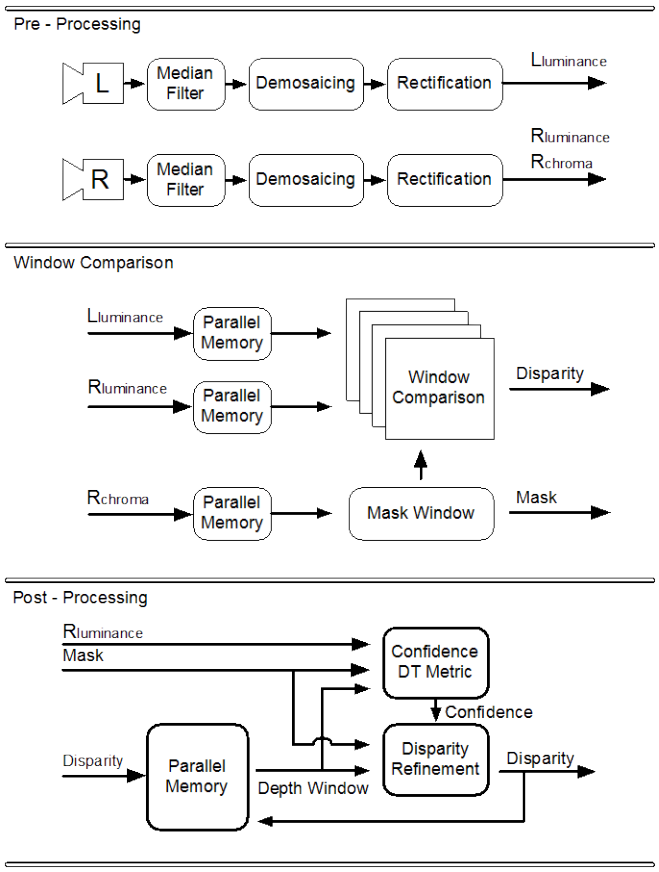


Fig. 2 Global architecture

One of the goals of this architecture is to limit the number of external components. Instead of the commonly used frame buffer to capture the pixels coming from the cameras, this architecture processes the pixel data without using a large buffer. One line buffer for each camera is needed to resolve the difference in clock speed between the camera and the memory write module (Fig. 3). A pre-processing module is placed just before the pixels entering the memory. Since the bit width of the data bus of the memory is commonly wider than the bit width of the pixel data, a multiplexer is used to combine successive pixel data in one memory write.

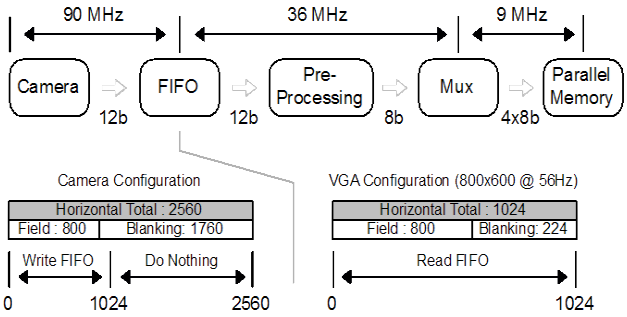


Fig. 3 Clock speed adaptation

The Pixel clock of the camera is configured to run at 90 MHz. Only the first 1024 pixels of each line are written to the FIFO. In this way the FIFO can be read out at 36 MHz without creating an underflow or overflow of the FIFO. The part of the blanking period that is written to the FIFO is the same size as the horizontal blanking period of the VGA specifications for a resolution of 800x600 with a frame rate of 56 Hz. In this way, the output of the FIFO can directly be connected to a VGA screen. The last step of this module stores four successive pixels in the parallel memory at a clock rate of 9MHz.

The two input streams of the cameras are compared with each other using a segmentation based Sum of Absolute Differences (SAD) calculation (Fig. 4). During every clock cycle a window of the right camera is compared with four windows of the left camera. Since four successive pixels are stored in one memory location, one memory read accesses four pixels, hence four comparison modules are running in parallel.

The frequency of the comparison module directly controls the possible depth range of the stereo matching architecture and can be adapted to the available resources. In the example on Fig. 4 a frequency of 180 MHz is chosen, which touches the design limit for implementation in a Cyclone II. In this case, the depth module runs twenty times faster than a memory write. This leads to a depth range of eighty when the comparison module has a depth of four.

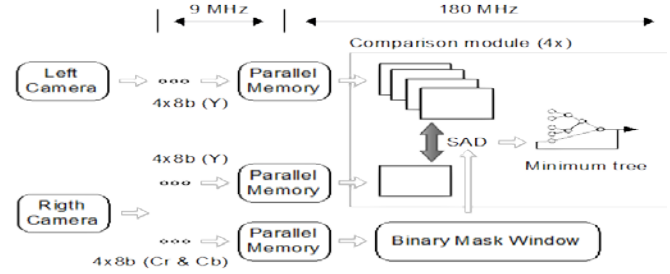


Fig. 4 Comparison module

The depth range of a single comparison module is limited to a small depth range. It can be increased to accommodate a larger depth range if the maximum operating frequency is reached. On each clock cycle (CC), the comparison module compares the reference window with four other windows (Fig. 5). The lowest SAD score and its corresponding index are saved in a register, so that on the next clock cycle this lowest SAD score can be compared against the SAD scores of the next four windows.

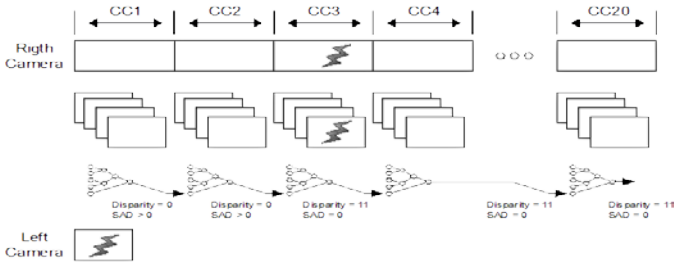


Fig. 5 Comparison module operating at a higher frequency

C. Pre-Processing

The pre-processing module consists of three different entities: first a median filter is used to remove defected pixels of the camera, next a demosaicing algorithm is used to reconstruct the color image and lastly a rectification module is used to remove lens distortion and perform stereo calibration (Fig. 6).

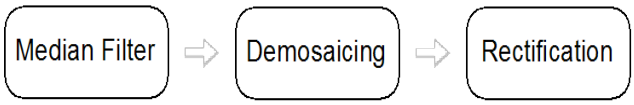


Fig. 6 Pre-processing module

Pixels coming out of the camera are formatted in a Bayer pattern consisting of the four colors: Red (R), Green1 (G1), Blue (B) and Green2 (G2), representing the three color filters (Fig. 7). The number of green pixels is 50% of the total amount of pixels, since the human eye is more sensitive to the color green.

R	G1	R	G1	R	G1
G2	B	G2	B	G2	B
R	G1	R	G1	R	G1
G2	B	G2	B	G2	B

Fig. 7 Bayer pattern

The median filter takes the median value of the pixels around the current pixel taking into account the Bayer pattern. An area of 5 x 5 is chosen around the current pixel, only the pixels with the same color are used to calculate the median value. Fig.8 shows the median filter where the pixels that contain the same color as the current pixel (P13) are shaded in gray. The median filter is a nonlinear filtering technique that preserves edges and removes defective pixels.

P1	P2	P3	P4	P5
P6	P7	P8	P9	P10
P11	P12	P13	P14	P15
P16	P17	P18	P19	P20
P21	P22	P23	P24	P25

Fig. 8 Median filter (current pixel = P13)

The demosaicing algorithm is used to estimate for every pixel the color levels for all color components at that pixel. Using linear interpolation, the missing RGB colors are reconstructed from the adjacent pixels (Fig. 9).

The proposed architecture makes use of the YCrCb color space. The Luminance (Y) values are used to compare the two input streams. While the chrominance values (Cr, Cb) are used to construct the binary mask window. Hence, the

reconstructed RGB color space needs to be transformed into the YCrCb color space (1).

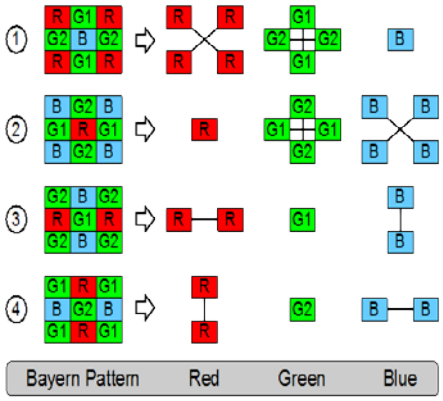


Fig. 9 Demosaicing

$$\begin{cases} Y = 16 + (66 \cdot R + 129 \cdot G + 25 \cdot B) \\ C_B = 128 + (-38 \cdot R - 74 \cdot G + 112 \cdot B) \\ C_R = 128 + (112 \cdot R - 94 \cdot G - 18 \cdot B) \end{cases} \quad (1)$$

Two different kinds of distortions are present in a stereo camera setup. The first one are the lens distortions, the second one is a misalignment of the two cameras (Fig. 10). Since the search space is only located on the epipolar line, both distortions should be resolved before the matching can be performed.

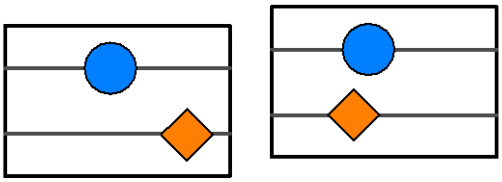


Fig. 10 Stereo camera misalignment (grey lines are the mismatched epipolar lines)

In order to remove these distortions, multiple images of a checkerboard pattern are taken using the camera setup and imported into Matlab. The intrinsic and extrinsic parameters of the cameras individually, and the transformation matrix of the stereo setup are calculated like proposed by Zhang [13]. These parameters are than used to construct the x and y mapping coordinates for each pixel in the image.

Storing the mapping coordinates for each pixel uses a large amount of memory. Since the mapping coordinates does not change drastically from pixel to pixel, it suffices to only store the mapping coordinates of certain pixels. These pixels are chosen to be located on a regular grid. The desired grid size depends on the amount of distortion in the image. A grid size of 33 x 21 has been chosen for this setup.

During the rectification process, bilinear interpolation is used to reconstruct the mapping coordinates for the complete image (Fig. 11). Formula (2) shows how the mapping coordinates for pixel 'p' (mapp) are calculated from the mapping coordinates of pixel 'a' (mapa), 'b' (mapb), 'c' (mapc) and 'd' (mapd), which are located on the rectangular grid.

This architecture limits the distance between grid points size to powers of two. Doing so enables the replacement of the division operators (2) by simple shift operators.

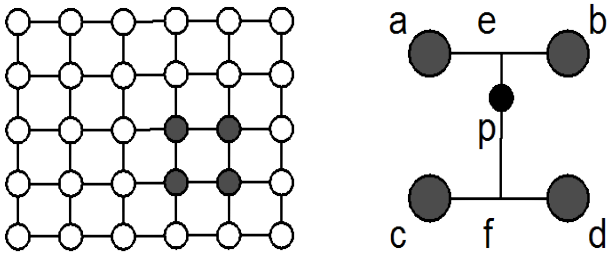


Fig. 11 Grid based mapping (left: complete grid, right: bilinear interpolation between grid points)

$$\begin{cases} map_e = (|ae| \cdot map_b + |eb| \cdot map_a)/|ab| \\ map_f = (|cf| \cdot map_d + |fd| \cdot map_c)/|cd| \\ map_p = (|ep| \cdot map_f + |pf| \cdot map_e)/|ef| \end{cases} \quad (2)$$

D. Memory Architecture

Due to the sequential way in which digital video data are presented, video signal processing architectures are traditionally built around line buffers. In the line buffers a number of the most recent scan lines are kept on-chip. Line buffers could be implemented as shift registers, but are currently efficiently implemented as on-chip memory blocks with dedicated addressing logic, such that they are used as FIFOs, typically one FIFO per scan line. At the outputs of the FIFOs, corresponding column pixels for the recent scan lines are accessed. These can then be stored in a shift register array with the size of the area of interest for window based video operations such as filtering, edge detection, sharpening, resampling etc.

However, the traditional scan line based FIFO architecture does not allow for a complete window refresh on each clock cycle. It also does not fully exploit the parallelism that is available with multiple on-chip memory blocks. In [14] the authors present a parallel System-on-Chip (SoC) memory architecture for a stereo vision system (Fig. 12). It allows for a parallel access to all pixels located in a chosen window of the image and enables a complete window refresh on each clock cycle.

During every clock cycle, exactly one on-chip memory block is written to. An Address Management Unit (AMU) is needed to keep track of which on-chip memory and which address to write to.

During every clock cycle, reading the on-chip memory contents for a chosen window is performed in parallel.

The window retrieved from the memories is not immediately usable for further processing. Although all window pixels are available at the same clock cycle, they still need to be reorganized in a way that the window orientation in the memory represents the window orientation on the image. Due to the addressing order of the AMU, the pixels contained in a memory window are scrambled in comparison with the image window.

For the architecture with a fixed window cost aggregation, only a column transformation is needed for the complete window. For the architecture with the binary adaptable window cost aggregation, a column transformation is needed for the complete window and a complete transformation is needed for the centre pixel.

Using this architecture a complete window refresh on each clock cycle is possible, which is used in this work to increase the depth range of a stereo matching algorithm.

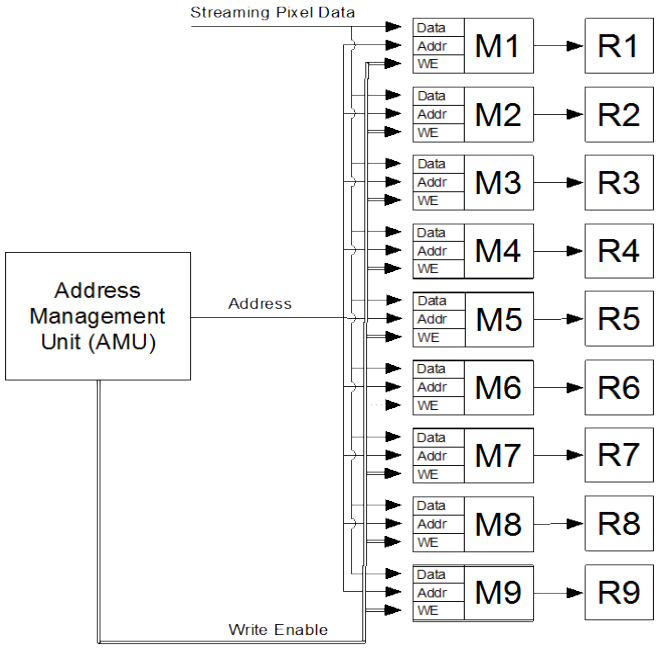


Fig. 12 Memory architecture

E. Binary Adaptable Window

When using a fixed window shape, implicitly depth continuity across this window is assumed. This assumption is not correct at depth edges, where the centre pixel depth is different from some (or the majority) of the surrounding pixels depths. A more conservative assumption is to assume depth continuity across pixels with similar color. The adaptive weight algorithm proposed by Yoon and Kweon [4] gives a support weight to each pixel in a fixed squared window. A derivative of this method is implemented in [5]. To save SoC resources, the support weights are modelled as binary values, where '0' means that this pixel will give no support to the matching window and '1' means that this pixel gives support to the matching window. This can be seen as a masking window that selects the pixels that will be used in the cost aggregation phase of the SAD algorithm. Only the pixels where the masking window is '1' (black) will be taken into account (Fig. 13).

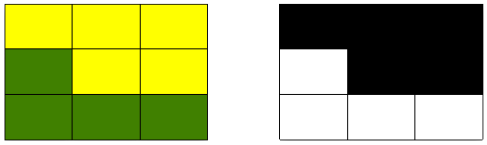


Fig. 13 Window content (left) and resulting window mask (right)

F. Cost Aggregation

The Sum of Absolute Differences (SAD) calculates the differences between two selected windows. It is a measure of similarity between two parts of an image. Fig. 14 shows the calculation of the main building block of the calculation; the absolute difference (AD).

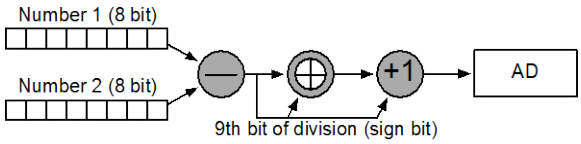


Fig. 14 Absolute difference calculation

The summation part of the SAD uses a window and depth parallel approach to calculate the sum of absolute differences for all depths and all windows in one clock cycle. As previously stated, only depth continuity across pixels with similar color is assumed. For each window a binary mask window is generated, which selects the supporting pixels in the cost aggregation phase of the SAD algorithm. This selection is performed by using a color similarity metric (3), where ΔCr and ΔCb are the absolute differences between the pixel in the window and the centre pixel.

$$w = \begin{cases} 0 \; if(\Delta Cr + \Delta Cb) > threshold \\ 1 \; otherwise \end{cases} \quad (3)$$

$$SAD = \sum_{i=1}^{window\ \ size} w * absolute\ difference(i) \quad (4)$$

Since the weights (w) in this architecture are '0'or '1', the multiplication in (4) can be replaced by an AND operator. This allows an efficient hardware implementation.

G. Minima Selection

The minima selection is based on an iterative minima tree calculation. The SAD results are pair wise compared (Fig. 15), while each time the lowest value is stored in a register. Afterwards, these registers are pair wise compared and stored in registers. These steps are repeated until one value remains. Storing of the intermediate results in registers makes this method interesting for pipelining. Four registers are needed which need to be large enough to hold the SAD value and their corresponding depth index. The result of the minima selection module is put in parallel memory architecture. Doing so will enable parallel access to all depth values in a chosen window for the following modules.

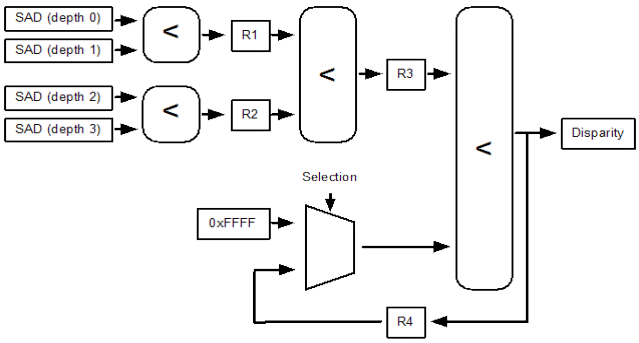


Fig. 15 Minima selection module

The selection input of the multiplexer decides if the previous results are incorporated or disregarded.

H. Post-Processing

The post-processing module consists of two separate entities: a decision tree based confidence metric module and a disparity refinement module (Fig. 16). Note that the post-processing module can be run on a higher frequency compared to the input stream, this allows for an iterative refinement of the depth map.

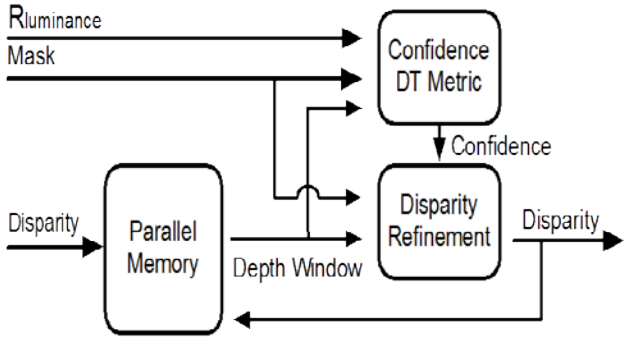


Fig. 16 Post-processing module

A selection of different confidence metrics have been combined to train a decision tree (DT) confidence metric as described in [9]. The decision tree is a top- down tree structure consisting of internal nodes, leaf nodes, and branches. Each internal node represents a decision on a feature, and each outgoing branch corresponds to a possible outcome. Each leaf node represents a class (0 or 1 in this case). The selected features for input to the decision tree are the following:

The texture (TEX) uses a fixed window of luminance information (5) around the investigated pixel and measures the amount of texture it contains. The intuition behind it is that textureless regions will result in more incorrect depth values.

$$TEX = max_{window}(Y_i) - min_{windo\,w}(Y_i) \quad (5)$$

The "Sum of Neighboring Depths Differences Binary Window" (SNDDBW) calculates the depth differences between the center pixel and its neighboring pixels, which have a similar color. This is a different usage of the binary mask window presented in [5]. The assumption is to have only depth continuity across pixels with similar color.

$$w = \begin{cases} 0 \; if(\Delta Cr + \Delta Cb) > threshold \\ 1 \; otherwise \end{cases} \quad (6)$$

$$SNDDBW = \frac{\sum_{i=1}^{window\ \ size} w*|D_1(i) - D_1(center)|}{\sum_{i=1}^{windo\ \ w\ size} w} \quad (7)$$

The window size and the chroma threshold is chosen to be the same as in the cost aggregation phase such that the calculated mask window (1) ,which is available from the cost aggregation module, can be reused. After training and pruning of the DT, the resulting decision tree classifier can be seen in Fig. 17.

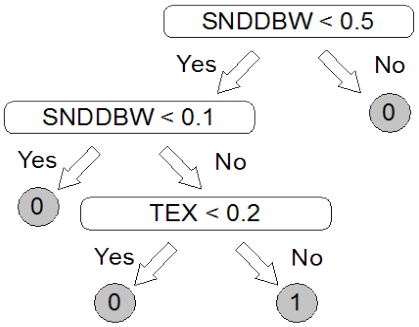


Fig. 17 Binary confidence DT metric

For every depth value that is labelled as unreliable, the disparity refinement step calculates a new value. This new depth value is calculated from the mean of the depth values of all neighbouring pixels, which have a similar color (8).

$$D = \begin{cases} D \; if \; confidence == 1 \\ mean_w(D) \end{cases} \quad (8)$$

The selected neighbouring pixels are chosen to be the same as in the cost aggregation phase such that the calculated mask window (3) needed to select similar color pixels does not need to be recalculated.

IV. IMPLEMENTATION AND RESULTS

Matlab has been used to generate, out of the chosen parameters and the high level architecture, a complete stereo matching architecture for both simulation and hardware generation. This allows an initial check of the stereo matching architecture in Matlab before implementation on the actual hardware. Using this framework, comparison between different stereo matching parameters and architectures can be rapidly performed.

The architecture and methods presented in this paper have been implemented on an FPGA system, based on an Altera Cyclone II with 68,416 logic elements and 250 memory blocks. The sources of the input streams are two cameras with a resolution of 800x600 and a frame rate of 56 Hz. The current implementation consists of a 7x7 binary adaptive window SAD with a depth range of 80, a DT confidence metric and a disparity refinement module.

The architecture has been constructed to reduce memory usage. Hence there is no need for external memories. The reduction of external memory usage has the additional advantage that the latency between input frame and output frame becomes minimal. This makes this system suitable to be incorporated in real-time control loops.

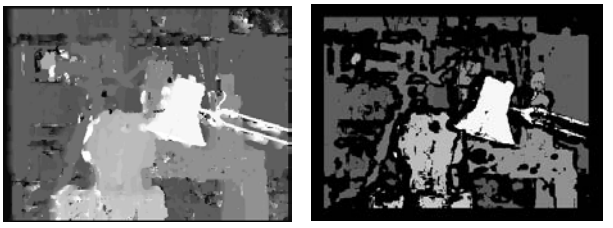
The current implementation is tailored to accommodate mainstream FPGA devices like the Altera Cyclone or Xilinx Spartan series without the need to use external memories. This allows for a true system on chip where only one chip is needed to provide the proposed functionalities.

Table 1 shows the resource consumption of the main blocks presented in this paper. The synthesis results are obtained using Quartus II 11.0 for an Altera Cyclone II-70.

TABLE I
RESOURCE USAGE FOR A 7X7 BINARY ADAPTIVE WINDOW SAD WITH A DEPTH RANGE OF 80, A DT CONFIDENCE METRIC AND A DISPARITY REFINEMENT MODULE.

	Module Name	#	Logic Elements		Memory Blocks	
			Single	Total	Single	Total
Pre - Processing	Median Filter	2	613	1,226	6	12
	Demosaicing	2	176	352	3	6
	Rectification	2	840	1,680	15	30
Window Comparison	Memory Left Y	1	3,344	3,344	28	28
	Memory Right Y	1	825	825	21	21
	Memory Right C	1	825	825	21	21
	Adaptable Window	1	2,641	2,641	0	0
	Cost Aggregation	16	1,707	27,312	0	0
	Minima Selection	4	95	380	0	0
Post - Processing	Parallel Memory	1	825	825	21	21
	DT Confidence Metric	1	1,788	1,788	0	0
	Disparity Refinement	1	1,216	1,216	0	0
Total				**42,414**		**139**

Fig. 18 shows the depth maps of the Tsukuba stereo pair [2] without confidence measurement compared with the DT confidence metric and after disparity refinements. The results indicate that the quality of the resulting depth map increases when using the disparity refinement steps. It can also be seen that depth mismatches not detected by the binary confidence DT metric will enlarge during the disparity refinement steps. This is because the disparity refinement module is not robust against larger groups of mismatches or against mismatches isolated from correct matches. Future implementations will address this problem by improving the classification rate of the confidence metric, and by making use of a more advanced disparity refinement step.



a. Without confidence metric b. With DT confidence metric

c. Disparity refinement (1 iteration) d. Disparity refinement (5 iterations)

Fig. 18 Depth map quality of the Tsukuba Stereo Pair [2] for a 7x7 binaire adaptive window (black pixels indicate a confidence of zero)

V. CONCLUSIONS

In this paper a real-time stereo vision SoC architecture for a depth-field generation processor is presented. It allows for a reduction of memory and logic resource requirements. The dense stereo problem is divided into two parts:

First a rough depth map is constructed using a segmentation based SAD window comparison. The pixels coming from the camera pass a pre-processing module consisting of a median filter, demosaicing unit and a rectification unit. Next, the two input streams are compared using a segmentation based SAD Winner Takes All (WTA) algorithm which outputs the depth value.

Second a disparity refinement module identifies false matches using a binary confidence metric and replaces them with new estimates derived from the neighbouring depth values.

It is shown that addition of the post-processing module increases significantly the quality of the depth map.

ACKNOWLEDGMENT

Research sponsored in part by BOF (Bijzonder Onderzoeksfonds) UHasselt, Flanders FWO (Fonds voor Wetenschappelijk Onderzoek) and Chinese MOST (Ministry of Science and Technology) project number G.A.063.10.

REFERENCES

[1] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: From software to hardware," International Journal of Optomechatronics, vol. 2 (4), 2008, pp. 435-462

[2] D. Scharstein, and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International Journal of Computer Vision, vol. 47 (1), 2002, pp. 7-42.

[3] R. Zabih, J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence", in Proc. European Conference on Computer Vision, Stockholm, Sweden, May 1994, pp. 151-158.

[4] K.J. Yoon, and I.S. Kweon, "Adaptive support-weight approach for correspondence search," IEEE Trans. PAMI, vol. 28 (4), 2006, pp. 650-656.

[5] Motten, L. Claesen, "A Binary Adaptable Window SoC Architecture for a Stereo Based Depth Field Processor," in Proceedings IEEE VLSI-SOC-2010, 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, Madrid, 27-29 September 2010, pp. 25 - 30.

[6] K. Ambrosch, M. Humenberger, and W. Kubinger, "SAD-based stereo matching using FPGAs," in Embedded Computer Vision, Advances in Pattern Recognition, K. Branislav, ed. London: Springer-Verlag, 2009, pp. 121-138.

[7] Yi, J. Kim, L. Li, J. Morris, G. Lee, and P. Leclercq, "Real-time three dimensional vision," in Advances in Computer Systems Architecture, Lecture Notes in Computer Science, Berlin: Springer-Verlag, vol. 3189 2004, pp. 309-320.

[8] X. Hu, P. Mordohai, "Evaluation of stereo confidence indoors and outdoors," in Proceedings IEEE CVPR-2010, 23th IEEE conference on Computer Vision and Pattern Recognition, 2010, pp. 1466-1473.

[9] Motten, and L. Claesen, "Binary confidence evaluation for a stereo vision based depth field processor SoC," in Proceedings IEEE ACPR-2011, 1st Asian Conference on Pattern Recognition, Beijing, 28-30 November 2011, 5 pages (accepted for publication).

[10] Murphy, D. Lindquist, A. M. Rynning, T. Cecil, S. Leavitt, and M. L. Chang, "Low-cost stereo vision on an FPGA," International Symposium on Field-Programmable Custom Computing Machines, 2007, pp. 333-334.

[11] M. Hariyama, and M. Kameyama, "Pixel-serial and window-parallel VLSI processor for stereo matching using a variable window size," Interdisciplinary Information Sciences, vol. 7 (2), 2001,pp. 289-297.

[12] S. Jin, J. Cho, X. D. Pham, K.M. Lee, S. –K. Park, and J. W. Jeon, "FPGA Design and Implementation of a Real-Time Stereo Vision System," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20 (1), 2010, pp. 15-26

[13] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," in Proceedings IEEE ICCV-1999, 7th IEEE International Conference on Computer Vision, Kerkyra, 20-25 September 1999, pp. 666 - 673.

[14] Motten, L. Claesen, "An On-Chip Parallel Memory Architecture for a Stereo Vision System," in Proceedings IEEE ECECS-2010, 17th IEEE International Conference on Electronics, Circuits, and Systems, Athens, 12-15 December 2010, pp. 500 - 503.